Contents

Traffic Shifting

3 minute read page test

Before you begin

Apply weight-based routing

Understanding what happened

Cleanup

See also

This task shows you how to shift traffic from one version of a microservice to another.

A common use case is to migrate traffic gradually from an older version of a microservice to a new one. In Istio, you accomplish this goal by configuring a sequence of routing rules that redirect a percentage of traffic from one destination to another.

In this task, you will use send 50% of traffic to `reviews:v1` and 50% to `reviews:v3`. Then, you will complete the migration by sending 100% of traffic to `reviews:v3`.

Before you begin

- Setup Istio by following the instructions in the Installation guide.
- Deploy the Bookinfo sample application.
- Review the Traffic Management concepts doc.

Apply weight-based routing

If you haven’t already applied destination rules, follow

the instructions in Apply Default Destination Rules.

1. To get started, run this command to route all traffic to the `v1` version of each microservice.

```
$ kubectl apply -f @samples/bookinfo/networking/virtual-service-all-v1
.yaml@
```

2. Open the Bookinfo site in your browser. The URL is `http://$GATEWAY_URL/productpage`, where `$GATEWAY_URL` is the External IP address of the ingress, as explained in the Bookinfo doc.

   Notice that the reviews part of the page displays with no rating stars, no matter how many times you refresh. This is

because you configured Istio to route all traffic for the reviews service to the version `reviews:v1` and this version of the service does not access the star ratings service.

3. Transfer 50% of the traffic from `reviews:v1` to `reviews:v3` with the following command:

```
$ kubectl apply -f @samples/bookinfo/networking/virtual-service-reviews-50-v3.yaml@
```

   Wait a few seconds for the new rules to propagate.

4. Confirm the rule was replaced:

```
$ kubectl get virtualservice reviews -o yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
...
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
```

5. Refresh the `/productpage` in your browser and you now see *red* colored star ratings approximately 50% of the time.

This is because the `v3` version of `reviews` accesses the star ratings service, but the `v1` version does not.

With the current Envoy sidecar implementation, you may need to refresh the `/productpage` many times –perhaps 15 or more–to see the proper distribution. You can modify the rules to route 90% of the traffic to `v3` to see red stars more often.

6. Assuming you decide that the `reviews:v3` microservice is stable, you can route 100% of the traffic to `reviews:v3` by applying this virtual service:

```
$ kubectl apply -f @samples/bookinfo/networking/virtual-service-reviews-v3.yaml@
```

Now when you refresh the `/productpage` you will always see book reviews with *red* colored star ratings for each review.

Understanding what happened

In this task you migrated traffic from an old to new version of the `reviews` service using Istio's weighted routing feature. Note that this is very different than doing version migration using the deployment features of container orchestration platforms, which use instance scaling to manage the traffic.

With Istio, you can allow the two versions of the `reviews` service to scale up and down independently, without affecting the traffic distribution between them.

For more information about version routing with autoscaling, check out the blog article Canary Deployments using Istio.

Cleanup

1. Remove the application routing rules:

```
$ kubectl delete -f @samples/bookinfo/networking/virtual-service-all-v1.yaml@
```

2. If you are not planning to explore any follow-on tasks, refer to the Bookinfo cleanup instructions to shutdown the application.